Massachusetts Institute of Technology
C. S. Draper Laboratory
Cambridge, Massachusetts

LUMINARY Memo #139

To: Distribution
From: D. Eyles
Date: 3 March 1970
Subject: Description of Variable Servicer

The Variable Guidance Period Servicer is now running in ZERLINA revision 3. This is the Servicer and the version -- introduced in Luminary Memo 138 -- inspired by the fact that LM powered flight is approaching a sort of asteroid belt of TLOSS difficulties as its time margin slips away. The new Servicer is performing, as expected, very well. It is time to describe it in greater detail.

The new Servicer is characterized by structural simplicity in return for a little extra algebra to replace assumptions about guidance period.

The structural changes are the most important. The READACCS task has disappeared, though the tag survives, and the accelerometers are read in the Servicer job. The rigid timing of which tasks are capable is not needed since Servicer computes guidance period and does not care exactly when the PIPAs are read. The Servicer job is no longer set up at fixed intervals by READACCS; it is continuously running. When it finishes, after Guidance puts out its commands and displays, it simply starts over. Because Servicer is only one long job, not a series of jobs, it can never overlap itself. Thus Servicer can never clobber its own erasables (as it can now), and it is impossible for tail-ends of self-interrupted Servicer jobs to hang around in the vac areas and core sets (as they can now), liable to be executed long after they apply with pernicious results, such as the issuing of anachronistic commands. No job's exact timing is important. With the new Servicer we do not care when a given job runs or how long it takes. (At most we require it to run in a one-to-one correspondence with Servicer passes.) We have only to make sure that every essential job does get done, and this is easily accomplished with a priority system in which Servicer has the lowest

priority of all the jobs that must run. Servicer runs at priority 20. With the exception of a few extended verbs, all the other jobs which may be set up while Servicer is running are given higher priorities so they will break in on it. This is true of present ropes too. The only jobs Servicer could lock out are extended verbs, like V82, which lower their priority from the initial 30 to something below Servicer's. The variable Servicer will occasionally have to go to sleep to give these time to function. There are three or more satisfactory ways to do this of which the simplest -- to use the Minimum Period Logic (described below) -- is the one implemented in ZERLINA. Incidentally, the landing radar read, which used to include off-line jobs LRHJOB and LRVJOB, is now done entirely in interrupts, starting with R12READ; this is true of current LUMINARYs as well as ZERLINA. The other Servicer auxilliary, R10, R11, known as QUARTASK to signify its nature, also uses interrupts.

By the elimination of READACCS and the adoption of the Cyclical PIPA Reader (described below) particularly, Servicer's restart protection is also simplified. Group 5 is used for Servicer itself and for Guidance which together make up the Servicer job. As written in ZERLINA, Servicer does without restart tables. A specialized "phase change" routine called SERVCHNG is provided in fixed-fixed (7 words) for use by Servicer and Guidance in setting restart points. Both LASTBIAS (set up by PREREAD) and 1/PIPA (called by Servicer proper) which in turn sets up 1/GYRO are skipped rather than repeated in the event of a restart while they are in progress. This has always been true although it is a complication and the reason, unless we really expect the instruments to behave better rather than worse than their compensation, is not apparent. Group 3 is used by BURNBABY (along with groups 1, 4 and 6) and for P66ROD. P66ROD can no longer be half-in half-out of the Servicer job as it was. It becomes a higher priority job (24) running at regular intervals and asynchronous with Servicer. P66ROD will also be more thoroughly restart protected than it was before. P66HZ, the attitude command half of P66, stays in the Servicer job. QUARTASK, vital because besides the Landing Analog Displays (R10) it contains the Abort Monitor (R11), is rigorously protected in group 2. Individual landing radar reads cannot be restart protected practically and never have been for it is sufficient to protect the routine that sets them up,

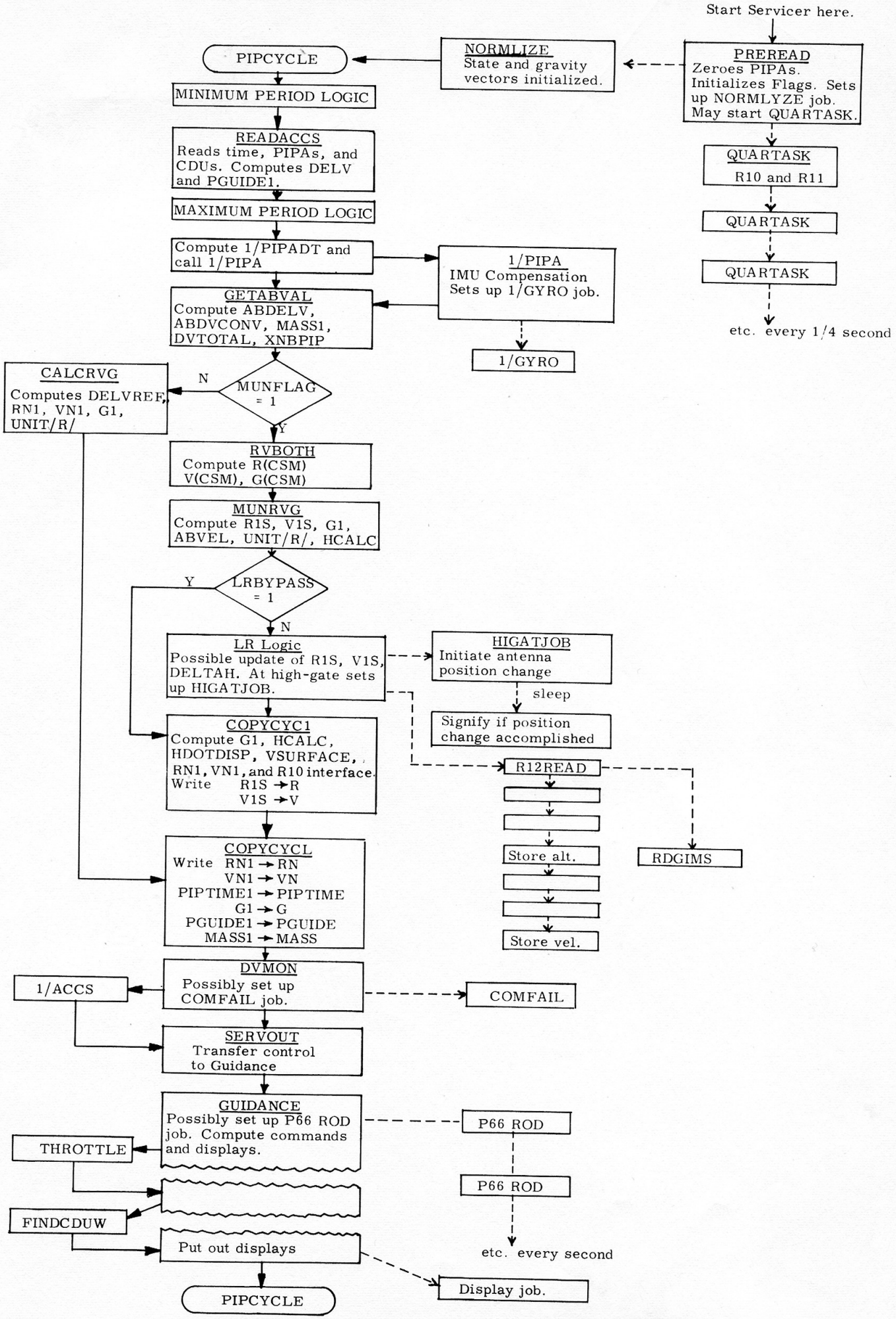
namely Servicer.

In words the new Servicer compares as follows with LUMINARY 1C and 1D:

	LUMINARY 1C	LUMINARY 1D	ZERLINA
bank 22	24	23	
bank 23	24	24	
bank 27	31	31	
bank 32	25	3	
bank 33	920	808	1019
bank 34	77		
bank 37	162	171	77
	1263	1060	1096

Although this comparison is somewhat biased against LUMINARY 1C by the fact that it included in Servicer some landing radar routines which in 1D and ZERLINA are replaced by added coding in another section, it is still true that the new variable Servicer takes more words than the fixed Servicer in 1D but less than the one in 1C.

On the next page is a rough picture of the new Servicer job to make the description that follows a little clearer.


Start Servicer here.
PREREAD
Zeroes PIPAs. Initializes Flags. Sets up NORMLYZE job. May start QUARTASK.
NORMLIZE
State and gravity vectors initialized.
PIPCYCLE
QUARTASK
R10 and R11
QUARTASK
QUARTASK
etc. every 1/4 second
MINIMUM PERIOD LOGIC
READACCS
Reads time, PIPAs, and CDUs. Computes DELV and PGUIDE1.
MAXIMUM PERIOD LOGIC
Compute 1/PIPADT and call 1/PIPA
1/PIPA
IMU Compensation Sets up 1/GYRO job.
GETABVAL
Compute ABDELV, ABDVCONV, MASS1, DVTOTAL, XNBPIP
1/GYRO
CALCRVG
Computes DELVREF, RN1, VN1, G1, UNIT/R/
N
MUNFLAG = 1
Y
RVBOTH
Compute R(CSM) V(CSM), G(CSM)
MUNRVG
Compute R1S, V1S, G1, ABVEL, UNIT/R/, HCALC
Y
LRBYPASS = 1
N
LR Logic
Possible update of R1S, V1S, DELTAH. At high-gate sets up HIGATJOB.
HIGATJOB
Initiate antenna position change
sleep
Signify if position change accomplished
COPYCYC1
Compute G1, HCALC, HDOTDISP, VSURFACE, RN1, VN1, and R10 interface.
Write R1S → R
V1S → V
R12READ
Store alt.
Store vel.
RDGIMS
COPYCYCL
Write RN1 → RN
VN1 → VN
PIPTIME1 → PIPTIME
G1 → G
PGUIDE1 → PGUIDE
MASS1 → MASS
DVMON
Possibly set up COMFAIL job.
1/ACCS
COMFAIL
SERVOUT
Transfer control to Guidance
GUIDANCE
Possibly set up P66 ROD job. Compute commands and displays.
P66 ROD
P66 ROD
THROTTLE
FINDCDUW
Put out displays
etc. every second
Display job.
PIPCYCLE

PIPCYCLE begins the Servicer cycle. NORMLIZE, the Servicer initialization routine and lead in of the Servicer job (set up by PREREAD which is called by the TIG -30 task of BURNBABY) falls through to PIPCYCLE. And it is to PIPCYCLE that Guidance returns when it finishes. But PIPCYCLE is just a tag and a restart point.

First to be executed after PIPCYCLE is the Minimum Period Logic: if it has been less than PGMIN since the last accelerometer reading, then Servicer goes tosleep for the remaining time. The reasons for such logic are to establish a (hard) lower bound on guidance period for scaling purposes, because PGUIDE is sometimes in the denominator, and to provide a mechanism for granting (but not guaranteeing since if there is TLOSS this time is used up before period stretches) some time to low priority extended verbs, and for insuring enough time for at least one full downlist to be sent for each Servicer cycle. Two seconds is the obvious choice for PGMIN because that is how long it takes to send a downlist. (For P40s cases where the downlink data may be less urgently required an extended verb could be provided to shrink the minimum to one second; by speeding up the computation and display loop this might save a little RCS fuel in nulling residuals.) Two seconds has the second advantage of making the new Servicer's timing very like that of the one it replaces in the absence of enough TLOSS or program activity to stretch it longer. This would serve to maximize initial confidence in variable Servicer ropes.

Next, after a restart point and under inhint, come the reading of time into PIPTIME1, the reading of the PIPAs into the PIPATMPs and the CDUs into the CDUTEMPs, and the computation of DELV and PGUIDE1. PGUIDE1 is the length of the PIPA interval scaled in units of 2^{14} centiseconds with its lower order half always zero. The PIPAs are read according to a recent philosophical advance called the Cyclical PIPA Reader: after PREREAD the PIPAs are never zeroed and each DELV is computed, not read from the PIPA, as the difference between this and the previous value of the PIPA, stored in PIPAOLD. Suitable correction for the chance that the PIPA in question overflowed between the two readings is easily performed under the assumption that 81.92 m/s cannot be accumulated during one PIPA interval.

This won't happen until the LM gets a J-2 or PGUIDE reaches 10 seconds. As Al Harrano demonstrated to me, the PIPAs behave with perfect grace when they overflow. No counts are lost. One good thing about the Cyclical PIPA Reader is the straightforwardness it gives the PIPA reader restart protection. If there is a restart while it is in progress the clocks and PIPAs are reread and DELV is recomputed; for a PIPA reader like the present one which destroys the contents of the PIPAs the problem is much hairier. Second, because interfacing routines like P66ROD and the Landing Analog Displays were hampered by Servicer's periodic flushing of the PIPAs, the new scheme will assist them. By putting the reading of the PIPATMPs into the PIPAOLDs in COPYCYCL we enable off-line routines, asynchronous or otherwise, to integrate from R and V at PIPTIME to the current time in a consistent way by always using the current PIPAs-PIPAOLDs as the thrust delta-V, suitably corrected for possible PIPA overflow by a subroutine which is provided. The Cyclical PIPA Reader is one of the ideas that make it easier to write a variable Servicer, but because it is an interesting notion in its own right I'll probably describe it in more detail in a future memo.

After another restart point comes the Maximum Period Logic. Here if PGUIDE1 exceeds PGMAX an ALARM is issued, tentatively number 555. This logic establishes a (soft) upper limit on guidance period for scaling purposes. Three or four seconds would be a reasonable PGMAX. (Note that both PGMAX and PGMIN could be put in erasable if there were ever a reason to.)

Next 1/PIPADT is computed from PGUIDE1 and 1/PIPA is called. 1/PIPA in turn sets up 1/GYRO (at priority 21) as an off-line job. The scaling of 1/PIPADT is changed from 2^8 cs. to 2^{10} cs. to accomodate guidance periods greater than 2.55 seconds. This scale change involves altering 3 words in 1/PIPA and 1/GYRO.

From this point on the variable Servicer is little changed from the present one.

Unchanged are the computations that begin at GETABVAL, after another restart point, of ABDELV, ABDVCONV, MASS1, DVTOTAL, and XNBPIP, the body-platform matrix valid for PIPTIME1. The quantities ABDELV, ABDVCONV, and DELV are strictly delta-Vs, not accelerations, and it is no longer valid to use them as accelerations unless PGUIDE enters the computation. This means small modifications in 1/ACCS, DVMON,

THROTTLE and Ascent Guidance. Because DELV is in the downlink possibly RTCC programs will require a small change here.

Next come the average-G integration routines. In P40s cases near Earth or Moon CALCRVG is used. Descent, ascent, and the aborts use RVBOTH. At RVBOTH the CSM position, velocity and gravity vectors for PIPTIME are computed, stored in temporaries, and after a restart point copied into R(CSM), V(CSM) and G(CSM). Then, after another restart point, the LM state is integrated. These three average-Gs were modified to take account of guidance period and to compute G1, a gravitational acceleration in units of 2^{-6} m/cs^2 (full scale 156.25 m/s^2), instead of GDT1/2 which assumed a two second guidance period. The scaling of G1 and G and PGUIDE1 and PGUIDE is such that multiplied they match the extinct GDT1/2 and GDT/2 exactly, facilitating the changeover. As a matter of fact GDT/2 was a contrivance that at most saved a couple of words in the average-G computations. G suits most users better. The average-G alterations are navigationally the core of the variable Servicer, and were well tested a year ago in version DIANA by MOONLIGHT.
The equations as currently programmed are:

PGUIDE = 2 seconds

$\overline{R}1S = PGUIDE\ (\overline{V} + \overline{D}ELV/2 + \overline{G}DT/2) + \overline{R}$

Compute $\overline{G}DT1/2$ from $\overline{R}1S$.

$\overline{V}1S = \overline{G}DT1/2 + \overline{D}ELV/2 + \overline{G}DT/2 + \overline{D}ELV/2 + \overline{V}$

As programmed in ZERLINA these become:

PGUIDE1 = PIPTIME1 - PIPTIME

$\overline{R}1S = (PGUIDE1\ \overline{G}/2 + \overline{D}ELV/2 + \overline{V})PGUIDE1 + \overline{R}$

Compute $\overline{G}1$ from $\overline{R}1S$.

$\overline{V}1S = PGUIDE1\ \overline{G}1/2 + PGUIDE1\ \overline{G}/2 + \overline{D}ELV/2 + \overline{D}ELV/2 + \overline{V}$

This change took no words.

Following RVBOTH, unless LRBYPASS is set, the landing radar incorporation logic is executed. CALCRVG goes straight to COPYCYCL via a restart point. Changes in the landing radar logic include the addition of an average-G integration to find LM altitude and velocity at the time of the radar read, since R12READ is no longer synchronized with the PIPA reading. It could be, with difficulty, but it seems preferable to perform

the integration and remain free of all timing constraints. In addition a Nav Base to Stable Member transformation must be performed on the beam vectors which is valid at the time of the read rather than at PIPTIME. RDGIMS, a task set up by R12READ, records PIPAs, CDUs and time at the mid-point of the read. These changes are almost like putting back coding which existed in LUMINARY 1C and earlier when the velocity read was unsynchronized. At the end the incorporation logic clears the measurement-made discretes and sets up R12READ which is performed during the rest of Servicer and Guidance.

The miscellaneous computations at COPYCYC1 are virtually unchanged except for the addition of a recalculation of G1. This is needed because a G1 valid at radar read time is left by the LR logic, and because the LR logic may have updated the length of the position vector.

To the string of variables copied at COPYCYCL is added PGUIDE1 which is copied into PGUIDE. Also at COPYCYCL the PIPATMPs, which contain the PIPA value at PIPTIME, are written into the PIPAOLDs for use by the next Servicer cycle and by off-line routines.

DVMON is not changed (except as indicated above with reference to ABDELV) and after 1/ACCS is called control is transferred to the Guidance routine specified by the contents of AVGEXIT, just as always. If no Guidance is hooked up AVGEXIT contains the 2CADR of PIPCYCLE.

Changes in Guidance are as follows: (1) G must be used in place of GDT/2 to subtract away from total desired acceleration to get desired thrust acceleration. This typically saves a few words. (2) If ABDELV or ABDVCONV is used, as in ascent, a change may be necessary, as indicated above, to remove the assumption of a two second period. (3) Guidance must restart protect itself in a way consistent with Servicer in group 5. This is eased by the provision of SERVCHNG which does a variable type of "phase change" when called with a TC. (4) Guidance must raise its priority (to 23) before calling display routines so that the off-line jobs created have a higher priority than the Servicer job. On return from the display routine the priority has to be lowered again to 20. (5) The VACRLEAS calls must be eliminated from Ascent and Descent. (6) The throttle will have to be modified as indicated above and the other Guidance

output routine FINDCDUW will be modified in a simple way to prevent overshoot in the event of a stretched period. (7) Guidance must at the end transfer control to PIPCYCLE and avoid like poison going to ENDOFJOB, because this would terminate Servicer.

It is not impossible that in some ways the variable Servicer will not turn out precisely as described, but this is a description of an actual, running Variable Guidance Period Servicer which contains no known shortcomings (except for a DVMON which is an eyesore) and is so superior to the old one that if I were going to the Moon next week I would almost prefer to fly with ZERLINA than with LUMINARY.

Since drafting the above I have learned from Allan Klumpp of two even more elaborate ways for powered flight to go wrong due to TLOSS. Such possibilities seem likely to turn up at the rate of one or two a day for as long as Allan looks for them (although admittedly he is stress testing) -- and for as long as we preserve the present amusement park organization of Servicer and its adjuncts.